Remarks

This REQUEST FOR CONTINUED EXAMINATION and REPLY is in response to the Final Office Action mailed February 21, 2008. A Petition for Extension of Time is enclosed herewith, together with the appropriate fee. No fee is due for the addition of new claims.

I. Summary of Examiner's Rejections

Prior to the Office Action mailed February 21, 2008, Claims 1-7, 9, 11-17, 19, 21-27, 29 and 31-36 were pending in the Application. In the Office Action, all of the claims were rejected under 35 U.S.C. 103(a) as being unpatentable over Taylor et al. (U.S. Publication No. 2004/0019897, hereafter Taylor) in view of Sparks et al. (U.S. Publication No. 2003/0018950, hereafter Sparks).

II. Summary of Applicant's Amendments

The present Reply amends Claims 1, 2, 5, 7, 9, 11, 12, 15, 17 and 19; cancels Claims 21-27, 29 and 35-36; and adds new Claims 37-42, all as shown above, leaving for the Examiner's present consideration Claims 1-7, 9, 11-17, 19, 31-34 and 37-42. Reconsideration of the Application, as amended, is respectfully requested.

III. Claim Rejections under 35 U.S.C. §103

In the Office Action mailed February 21, 2008, Claims 1-7, 9, 11-17, 19, 21-27, 29 and 31-36 were rejected under 35 U.S.C. 103(a) as being unpatentable over Taylor (U.S. Publication No. 2004/0019897) in view of Sparks (U.S. Publication No. 2003/0018950).

Claim 1

Claim 1 has been amended to more clearly define the embodiment therein. As amended, Claim 1 currently defines:

> *1. (Currently Amended) A system for loading software applications, comprising:*
> *a server for storing and running a plurality of software applications, wherein each of said plurality of software applications includes a plurality of deployable modules and*

classes associated therewith, and wherein the software applications can be customized by a software developer and then deployed to run on the same server;

a control file, that can be edited by the software developer and associated with said plurality of software applications, wherein said control file specifies a hierarchy of application classloaders to be used with the modules in said plurality of software applications, and wherein the hierarchy includes a plurality of nested branches that are specified by the software developer to provide namespace separation between two or more of the plurality of software applications or between different modules in any one of the software applications; and

a deployment utility that, upon receiving a request to deploy and run a software application on the server, parses the control file and determines which classloaders are specified therein for the software application being deployed, and then loads with said software application onto the server a selection of said application classloaders corresponding to the hierarchy specified by said control file, including, if a particular software application or a module in a software application is being redeployed then loading only the application classloaders that are specified in the branches for that particular software application or module, without loading any of the other branches in the hierarchy.

In the Office Action, it was noted that the plain language of the claim did not exclude a server configured in a distributed form, and that the plain language of the claim did not appear to further limit or define the level of configuration file. Accordingly, Claim 1 has been amended to more clearly define that in the embodiment therein, the software applications can be customized by a software developer and then deployed to run on the same server; and that each of the plurality of software applications includes a plurality of deployable modules and classes associated therewith. Claim 1 has also been amended to more clearly define a deployment utility that, upon receiving a request to deploy and run a software application on the server, determines which classloaders are specified therein for the software application being deployed, and then loads with said software application onto the server a selection of said application classloaders corresponding to the hierarchy specified by said control file, including, if a particular software application or a module in a software application is being redeployed then loading only the application classloaders that are specified in the branches for that particular software application or module, without loading any of the other branches in the hierarchy.

The advantages of the embodiment defined by Claim 1 include that it provides more control over the classloader hierarchy, and allows the software developer to customize the classloader functionality, for example by placing modules frequently called by one another in the same branch within the hierarchy, and in the same search path, while at the same time allowing other modules to be separated into different branches and different application classloaders. The latter facilitates dynamic reloading, since upon receiving a request to deploy and run a software application on the server, the system determines which classloaders are specified therein for the software application being deployed, and then loads only the application classloaders that are specified in the branches for that particular software application or module.

Taylor discloses a method, system, and program for processing objects in a distributed computing environment. A determination is made of a program is needed to process a component. A file including the determined program is requested from a remote process. The requested file is received from the remote process, wherein the requested file includes a plurality of sections, each including different programs. At least one section includes programs that are intended only to be executed in a remote address space of the remote process and at least one other section includes programs that are intended to be downloaded from the remote process and execute in a client address space that is different than the remote address space. (Abstract). As further disclosed therein, the population manifest 326 is parsed by the container 306 to construct class loaders as necessary to load the components indicated in the manifests. FIG. 6 illustrates an example of a class loader hierarchy 350 that the population manifest 326 may implicitly define. The class loader hierarchy 350 divides class loaders into factory, root, and component class loaders. Component class loaders are used to load the variable components of the system, such as services, facility implementations, plug-ins, and dynamically loaded modules. Root class loaders load common interfaces that enable communication between the components of the system, such as facility interfaces. Factory class loaders are used to load factory classes. A factory is a component responsible for instantiating and destroying a particular type (or set of types) of components. A facility is a component whose instances can be shared across other components. (Paragraph [0050]). Each element in the one or more XML files forming the population manifest 326 indicates a component, encapsulated in a CAR file, to be loaded, and the attributes of the element may include information about the component, such as public package, version, etc. During runtime,

a new component may be added to the system by adding a new element to the XML file comprising a population manifest 326. A root XML file may maintain information on the root components that are loaded, where the last XML element in the root XML file would comprise the component that is loaded by the lowest root class loader in the hierarchy. (Paragraph [0052]).

Sparks discloses a server system further comprising a generic server interface configured to receive requests for a plurality of software modules; a module locator configured to interrogate at least a first software module of the plurality of software modules based on an identifier from a request received via the generic server interface, and to cause the first software module to be invoked based on the interrogation; and a redeployment interface configured to receive a redeployment request, and in response to the redeployment request, to cause the module locator to interrogate a second module instead of the first module based on the identifier. (Abstract). The original EJB/J2EE container model defines a contract between developers and application servers. In conventional systems, implicit in this specification is the notion that application server products must generate the code responsible for these additional service levels from the components supplied. (Paragraph [0019]). In a preferred embodiment, the [ ] invention uses an embedded Object Request Broker (ORB) to direct traffic to the correct EJB container and demarshal arguments. The container uses generic code to create the service levels required by the container. The use of dynamic skeletons means that no skeletons need to be generated to deploy new or modified application code. Components can therefore be deployed "as is" rather than forced through a series of code generation steps. (Paragraph [0023]-[0024]). In a preferred embodiment, the present server system manages class loaders, and can dereference an entire deployed application by dereferencing the class loaders associated with that application. This effectively un-deploys the application. By combining an un-deploy with a deploy of a new version, a redeploy suitable for iterative development results. All EJBs, servlets and other artifacts of an application are dereferenced, and the old class loader references are refreshed to point to new class loader instances. (Paragraph [0070]).

Applicant respectfully submits that, in Taylor, the system therein is used to support programs that are intended only to be executed in a remote address space of the remote process, and at least one other section includes programs that are intended to be downloaded from the remote process and execute in a client address space that is different than the remote address

space. Accordingly, Claim 1 has been amended as described above, to more clearly define that in the embodiment therein, the system comprises a server for storing and running software applications, wherein the software applications can be customized by a software developer and then deployed to run on the same server.

Applicant further respectfully submits that, in Taylor, the class loader hierarchy therein appears to use a manifest file at the application level. Similarly, Figures 7 and 8 appear to show that the class loader hierarchy stops at the population instance for the application, but does not appear to proceed to the module level within each population instance. Accordingly, Claim 1 has been amended as described above to more clearly define that in the embodiment therein, each of said plurality of software applications includes a plurality of deployable modules and classes associated therewith (examples of which include EJB, WAR, and other deployable modules that can be deployed as a unit).

Sparks appears to describe the need for a generic server interface to allow Enterprise Java Bean (EJB) servers not requiring the server to be restarted in order to load newly-modified EJB classes and replace the older versions of the classes. However, while Sparks apparently allows for the application source code to be modified, and then the application effectively undeployed and again re-deployed, this appears to be a way to assist the software developer in the iterative software development process, rather than a means of giving the software developer control over how the classloaders work, or which modules will be loaded into memory.

Applicant respectfully submits that neither Taylor nor Sparks appear to disclose a hierarchy of application classloaders that includes a plurality of nested branches that are specified by the software developer to provide namespace separation between two or more of the plurality of software applications or between different modules in any one of the software applications. Similarly, Applicant respectfully submits that neither Taylor nor Sparks appear to disclose a deployment utility that determines which classloaders are specified therein for the software application being deployed, and then loads with said software application onto the server a selection of said application classloaders corresponding to the hierarchy specified by said control file, including, if a particular software application or a module in a software application is being redeployed then loading only the application classloaders that are specified in the branches for that

particular software application or module, without loading any of the other branches in the hierarchy, as currently defined by Claim 1.

In view of the above comments, Applicant respectfully submits that Claim 1, as amended, is neither anticipated by, nor obvious in view of, the cited references, and reconsideration thereof is respectfully requested.

Claim 11

The comments provided above with respect to Claim 11 are hereby incorporated by reference. Claim 11 has been similarly amended by the current Reply to more clearly define the embodiments therein. For similar reasons as provided above with respect to Claim 1, Applicant respectfully submits that Claim 11, as amended, is likewise neither anticipated by, nor obvious in view of the cited references, and reconsideration thereof is respectfully requested.

Claims 2-7, 9, 12-17, 19, 21-27, 29 and 31-36

Claims 21-27, 29 and 35-36 have been canceled, rendering moot the rejection of these claims. Claims 2-7, 9, 12-17 and 19 depend from and include all of the features of either Claim 1 or Claim 11. Claims 2-7, 9, 12-17 and 19 are not addressed separately, but it is respectfully submitted that these claims are allowable as depending from an allowable independent claim, and further in view of the amendments to the independent claims and the comments provided above. Reconsideration thereof is respectfully requested.

IV. Additional Amendments

Claims 37-42 have been newly added by the present Reply. Applicant respectfully requests that new Claims 37-42 be included in the Application, and considered therewith.

V. Conclusion

In view of the above amendments and remarks, it is respectfully submitted that all of the claims now pending in the subject patent application should be allowable, and reconsideration thereof is respectfully requested. The Examiner is respectfully requested to telephone the undersigned if he can assist in any way in expediting issuance of a patent.

Enclosed is a PETITION FOR EXTENSION OF TIME UNDER 37 C.F.R. §1.136 for extending the time to respond up to and including June 30, 2008.

The Commissioner is authorized to charge any underpayment or credit any overpayment to Deposit Account No. 06-1325 for any matter in connection with this Reply, including any fee for extension of time, which may be required.

Respectfully submitted,

Date: 6/30/2008

By: _____

Karl F. Kenna
Reg. No. 45,445

Customer No.: 23910
FLIESLER MEYER LLP
650 California Street, Fourteenth Floor
San Francisco, California 94108
Telephone: (415) 362-3800